**Допълнителна домашна работа по Функционално програмиране**
**специалност „Информационни системи“, I курс, I група**
**2024/2025 учебна година**

---

Решенията трябва да са готови за автоматично тестване. Важно е програмният код да бъде добре форматиран. Предайте решенията на всички задачи в **един** файл с име **`hw3_<FN>.hs`**, където **`<FN>`** е Вашият факултетен номер.

Домашните работи се предават като изпълнение на съответното задание в курса по ФП в Moodle (https://learn.fmi.uni-sofia.bg/mod/assign/view.php?id=344494) най-късно до **28.05.2025 г. (сряда), 23:55**. Решения няма да могат да се предават след крайния срок. Ще се оценяват само файлове с разширение **`hs`**. Решения предадени като архив **няма** да се оценяват.

**Тази допълнителна домашна работа е за студентите от първа група, които не са успели да предадат решения на второто домашно или контролно.**

*Приятна работа и успех!*

---

**Задача 1 (25 точки).** Ще казваме, че едно цяло число без знак `k` е обратимо, ако сумата на `k` с числото `m`, образувано от същите цифри, но взети в обратен ред, се състои само от нечетни цифри.

Ако числото m започва с водеща нула, то считаме, че `k` не е обратимо.

Да се напише функция `reversibleNumbers :: Int -> [Int]`, която приема като аргумент число `n` и връща списък с всички обратими числа между `1` и `n` (включително).

*Примери:*

```
reversibleNumbers 20 → [12,14,16,18]
reversibleNumbers 31 → [12,14,16,18,21,23,25,27]
reversibleNumbers 10 → []
```

**Задача 2 (25 точки).** Нека е дефиниран алгебричен тип, представящ двоично дърво, както следва:

```
data BTree = Empty | Node Int BTree BTree deriving (Eq, Show)
```

Да се дефинира функция `calcProduct :: BTree -> Int -> Int`, която получава като параметри двоично дърво `bt` и цяло число `k`.

Функцията трябва да върне като резултат произведението на стойностите на тези върхове на `bt`, за които сумата на преките им наследници е по-голяма от `k`.

*Примери:*

```
bt = Node 5 (Node 1 (Node 5 (Node 1 Empty Empty)
                            (Node 2 Empty Empty))
                    (Node 2 (Node 3 Empty Empty)
                            (Node 4 Empty Empty)))
            (Node 2 (Node 1 (Node 1 Empty Empty)
                            (Node 7 Empty Empty))
                    (Node 9 (Node 5 Empty Empty)
                            (Node 2 Empty Empty)))

calcProduct bt 2 → 5 * 1 * 5 * 2 * 2 * 1 * 9 = 900
calcProduct bt 6 → 1 * 2 * 2 * 1 * 9 = 36
calcProduct bt 7 → 2 * 1 = 2
calcProduct bt 8 → 2
```